ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY (AAiT)

School of Information Technology and Engineering(SiTE)

MACHINE LEARNING AND BIG DATA

PROJECT THREE (CLUSTERING)

NAME: Natan Amanuel Mamo

Id no: ATR/7142/13

Year: IV (Software stream)

Submitted to: Mr. Bisrat

Table of Contents

INTRODUCTION

In this project, two methods of un-supervised learning, known as K-means clustering and Hierarchical clustering, were applied in this project to cluster movies in accordance to their features like genre, ratings, and other metadata etc. The primary goal of the study was to understand how clustering can reveal the perplexity that is enclosed in the given dataset without prior knowledge of cluster or relationship. This approach gives us the opportunity to divide movies into logical groups depending on some more or less descriptive signs.

First, the movies were introduced to the K-means clustering method to make the first partitioning and then the results were checked and optimized by hierarchical clustering method. The hierarchical clustering algorithm gave a finer structure which enabled the understanding of how variation in the clustering configurations affects the performance of the model. In our analysis section, having used the silhouette score to compare the two models of clustering, we were able to decide on the sort of quality of clusters that fit the dataset as well as decide on the number of clusters which was most appropriate.

The dataset used in this project is the Movie-Lens dataset, an extensively applied dataset which includes movie ratings and its related general information and genres. This dataset offered a good material to work on to analyze and cluster movies for a better understanding of them. As it seen, the results of clustering algorithms provide us with the insights of groupings of movies, which, for example, can be useful at creating movie recommendation system.

This paper aims to document the lifecycle of the process ranging from data preprocessing, performing the K-means and hierarchical clustering and comprehensive assessment of the appropriate clustering configurations.

To observe the effects of the employed linkage methods on cluster formation, we paid extra attention to determining the best number of clusters and include metrics such as silhouette score. Analysis of the obtained values subdivided the surveyed items into meaningful groups, which facilitated the identification of the optimal configuration of the number of clusters for interpretation.

In this report, the general approach to data processing and feature scaling, as well as the application of clustering is described. It also demonstrates how to assess other cluster configurations and why the best number of clusters should be chosen. The findings pointed to decision-level and feature-level adaptations when using hierarchical clustering and provided suggestions for future enhancements to use of the concept.

TOOLS AND METHODS

About the Dataset

The dataset used in this project is the MovieLens dataset, which is a widely used source of data for movie recommendation studies. MovieLens has rich movie-related data including the information on movies and ratings, and information about user's preferences. Specifically, this dataset includes the following key components:

Movies Data: The movies data consists of information about the movies themselves, including:

- Movie ID: An identification number given to each of the movies.
- Title: The name of the movie.
- Genres: A list of comma separated genre relevant to the movie including the Action, Comedy, Drama and so on.

Ratings Data: The ratings data contains information about the ratings provided by users for each movie, including:

- User ID: This is a code that can only belong to one user in the entire system.
- Movie ID: The IMDB ID of the movie which is being rated.
- Rating: A quantitative appraisal by a particular user focused on a particular movie, normally anchored on a 1-5 scale.
- Timestamp: A time stamp attributed to the occasion when the particular rating was made.

Users Data: The users data provides demographic information about the users who rated the movies, including:

- User ID: A name or number that no other user can share with another user.
- Gender: The gender of the user.
- Age: The age of the user.
- Occupation: The occupation of the user.
- Zip Code: The zip code of the user.

In the present work, the emphasis is made on grouping the movies according to the attributes such as genres, etc., not on user-oriented information. The movies and ratings datasets are therefore more applicable to the clustering analysis since the two hold more details of the movies and how these are viewed by users.

- It includes numerous motion pictures, which makes for a fair distribution of the films around a limited number of clusters according to type.• It has a good number of records (thousands of movies and ratings) and can be considered as a good sample for a clustering.• Features in a dataset such as genres are categorical so they can suitably be transformed into use for clustering like the one hot encoding or the TF-IDF vectorization.icient number of

records (thousands of movies and ratings), making it a representative sample for clustering tasks.

- The dataset's features (such as genres) are categorical, making them suitable for transformation into a format that can be used for clustering, such as one-hot encoding or TF-IDF vectorization.

In the frame of this work, just the movies dataset with fields like the movie title, genres etc, was mainly utilized to investigate clustering models, whereas the ratings data was not integrated into the process of clustering.

Jupyter Notebook

I used Jupyter Notebook for my data analysis tasks in Python. Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It is widely used in data science and machine learning for its ability to support interactive data visualization and the integration of code, which facilitates exploratory data analysis and prototyping.

Algorithms Implemented

Clustering

Clustering is an **unsupervised learning** technique used to group similar data points together based on certain features or patterns in the data. The primary objective of clustering is to categorize data points such that data points within the same cluster are more similar to each other than to those in different clusters. This makes clustering an ideal approach for discovering hidden structures in datasets without requiring labeled data.

In this project, we implemented two clustering algorithms to group movies based on their attributes: **K-means** clustering and **Hierarchical clustering**. Each algorithm has its own strengths and is suited to different types of data and use cases.

K-means clustering

K-means clustering is a **partitioning-based** algorithm that divides the dataset into a predefined number of clusters, denoted by **K**. The process follows these steps:

1. **Initialization**: Randomly select **K** initial centroids (cluster centers).
2. **Assignment**: Assign each data point to the nearest centroid.
3. **Update**: Recalculate the centroids based on the mean of the points assigned to each cluster.
4. **Repeat**: Repeat the assignment and update steps until the centroids no longer change or a set number of iterations is reached.

The **K-means** algorithm is preferred for clustering because:

- **Efficiency**: It is computationally efficient, especially for large datasets, because it minimizes the variance within clusters.
- **Scalability**: K-means is scalable to large datasets and is widely used in applications where speed is important.
- **Simplicity**: It's easy to understand and implement, making it a good starting point for clustering tasks.

However, K-means has some limitations:

- It requires the number of clusters (**K**) to be predefined.
- It can struggle with clusters of irregular shape or varying sizes.
- It is sensitive to initial centroid placement, which can lead to different results for different initializations.

In this project, **K-means** was used as a first step to divide the movie dataset into an initial set of clusters based on their genre and attributes.

Hierarchical clustering

Hierarchical clustering is a **tree-based** clustering algorithm that builds a hierarchy of clusters, creating a dendrogram (a tree-like structure). It does not require the number of clusters to be predefined and works by either merging smaller clusters (agglomerative approach) or splitting larger clusters (divisive approach). In this project, we used the **agglomerative** approach, which follows these steps:

1. **Initialization**: Start with each data point as its own cluster.
2. **Merging**: At each step, merge the two closest clusters based on a distance metric (e.g., Euclidean distance).
3. **Repeat**: Continue merging until all points are in a single cluster or the desired number of clusters is achieved.

The **Hierarchical clustering** algorithm is preferred for the following reasons:

- **No Need to Predefine Clusters**: Unlike K-means, it does not require the number of clusters to be set in advance, making it flexible for various scenarios.
- **Dendrogram Visualization**: The dendrogram produced by hierarchical clustering provides a visual representation of how clusters are formed, helping to choose an appropriate number of clusters based on the structure of the data.
- **Capturing Nested Structures**: Hierarchical clustering can capture nested clusters, which K-means might miss.

However, it has some limitations:

- **Computational Complexity**: It can be slower than K-means, especially for large datasets, since it requires computing distances between all pairs of data points.
- **Scalability**: It doesn't scale as well as K-means for large datasets due to the complexity of the algorithm.

In this project, **Hierarchical clustering** was applied after **K-means** to refine the clustering results and validate the effectiveness of the groupings. By using different linkage methods (such as "average" linkage), we could assess how the algorithm grouped the movies in a more granular way.

Both algorithms were evaluated using the **silhouette score**, a metric that assesses how well-separated the clusters are. Each clustering method contributed to a better understanding of the movie dataset and its inherent structures, allowing us to explore the optimal number of clusters for grouping movies.

Performance Metrics

In order to measure the quality of the clustering models with respect to the clusters we have produced, silhouette score was used. The silhouette score is perhaps the most popular performance metric in clustering tasks and is used to assess the distinctiveness of clusters.

Silhouette Score

Silhouette score determine cohesion and separation where cohesion quantifies how close data of a particular cluster are to each other; whereas separation defines how distant data of a cluster are to data belonging to another cluster. It is then summed up with regards to all the different points or data, and the resulting sum is divided with the number of points, so as to arrive at the score for that group. It provides a value between -1 and +1:

+1: Such a score near 1 means that the data points are well grouped. These points are much nearer to other points within their own cluster than to any point outside their cluster, which makes the clusters fairly distinct.

0: This is a near 0 resulting in the ability to suggest that this data point is on the edge of the decision boundary between two clusters meaning little separation.

-1: A value near -1 means that the data point was placed in the correct cluster but the cluster is too large as it is closer to points in a distant cluster.

CODE BREAKDOWN

IMPORTING LIBRARIES

In this project, I imported several essential libraries to facilitate data manipulation and analysis. Below is a breakdown of each imported library:

- Pandas (`import pandas as pd`): Pandas is a powerful data manipulation library that provides flexible data structures, like Series and DataFrames, for complex data analysis tasks. It is widely used for reading, writing, and processing structured data efficiently, enabling operations like filtering, grouping, and aggregating.
- NumPy (`import numpy as np`): NumPy is a foundational library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, alongside a collection of mathematical functions to operate on these arrays. It excels in performing efficient numerical operations and handling linear algebra.
- Matplotlib (`import matplotlib.pyplot as plt`): Matplotlib is a plotting library that is used for creating static, interactive, and animated visualizations in Python. It provides functionalities to create various types of plots and charts, enabling data visualization to help interpret data better.
- Seaborn (`import seaborn as sns`): Seaborn is built on top of Matplotlib and simplifies the creation of attractive statistical graphics. It helps visualize complex datasets with simpler interfaces and provides enhanced visual appeal by improving the aesthetics of visualizations.
- Scikit-learn (`from sklearn.model_selection import **train_test_split**`): Scikit-learn is a comprehensive library for machine learning that offers simple and efficient tools for data mining and data analysis. The module imported is used for splitting datasets into training and testing sets, which is essential in evaluating the performance of machine learning models.
- from sklearn.preprocessing import **LabelEncoder, StandardScaler**: imports two classes from the sklearn.preprocessing module:
    - LabelEncoder: Used to convert categorical data (e.g., 'male', 'female') into numerical labels (e.g., 0, 1).
    - StandardScaler: Used to standardize features by removing the mean and scaling to unit variance.
- **from sklearn.cluster import KMeans**: This line imports the KMeans class from the sklearn.cluster module. KMeans is an algorithm for performing k-means clustering, a popular unsupervised learning technique for grouping data points into clusters based on their similarity.
- **From sklearn.cluster import AgglomerativeClustering**: This line imports the AgglomerativeClustering class from the sklearn.cluster module. AgglomerativeClustering implements hierarchical clustering, which builds a hierarchy of clusters by iteratively merging the closest pairs of clusters.

- **from scipy.cluster.hierarchy import dendrogram, linkage**: This line imports two functions from the scipy.cluster.hierarchy module: dendrogram: Used to visualize the hierarchical clustering results as a dendrogram. linkage: Performs hierarchical clustering and returns the linkage matrix, which represents the distances between clusters at different stages of the merging process.
- **from sklearn.decomposition import PCA**: This line imports the PCA class from the sklearn.decomposition module. PCA (Principal Component Analysis) is a dimensionality reduction technique that transforms the data into a new set of features called principal components, which capture the most important variance in the data.
- **from sklearn.metrics import silhouette_score**: This line imports the silhouette_score function from the sklearn.metrics module.

Each of these libraries plays a crucial role in the workflow of data analysis, ensuring the efficient manipulation, analysis, and visualization of the dataset.

READ AND ANALYZE DATA

1. Reading the CSV File (data = pd.read_csv()): This function from the Pandas library is used to read a CSV (Comma Separated Values) file. The resulting data is stored in a DataFrame. A DataFrame is a 2-dimensional labeled data structure with columns that can be of different types (similar to a table in a database or a spreadsheet).

2. Viewing the First Few Rows (data.head()): This function displays the first five rows of the DataFrame by default. It allows you to quickly glance at the data to understand its structure and content.

3. Getting Basic Information About the DataFrame (`data.info()`): This function provides a concise summary of the DataFrame, including the number of entries, the data types of each column, and the number of non-null values. This is helpful for understanding the structure of the data and identifying potential data quality issues.

4. Distribution of Movie Ratings

5. Display Lowest and Highest rated movie

6. Display Lowest and Highest rated movie using Bayesian average in both ascending descending order

7. Information about Genres

8. Display Genre frequency

PREPROCESS AND CLEAN DATA

genres = movies['genres'].str.get_dummies('|'): Split Genres into separate columns

movie_ratings = ratings.groupby('movieId')['rating'].mean().reset_index(): This line calculates the average rating for each movie in the 'ratings' DataFrame.

movie_data = movies.merge(movie_ratings, on='movieId'): This line merges the 'movies' DataFrame with the movie_ratings DataFrame based on the 'movieId' column. This combines information about movies (like title, genres) with their average ratings.

features = movie_data.drop(['movieId', 'title', 'genres'], axis=1): This line creates a new DataFrame features by dropping the 'movieId', 'title', and 'genres' columns from the movie_data DataFrame. This leaves only the relevant features (average rating and genre columns) for further analysis or modeling.

features.isnull().sum(), features.fillna(features.mean(), inplace=True): Check for null values and fill those null values with a mean value.

Normalizing Numerical Features: scaler = StandardScaler(): This line creates an instance of the StandardScaler class. This class is used to standardize the features by removing the mean and scaling them to unit variance.

TRAIN THE MODEL

This section covers the implementation of the two clustering algorithms, namely, K-means clustering and Hierarchical clustering.

K-Means clustering: This code implements the K-Means clustering algorithm to the data, creates 3 clusters.

1. kmeans = KMeans(n_clusters=3, random_state=42): This line creates a KMeans object with the following parameters:

2. movie_data['cluster'] = kmeans.fit_predict(scaled_features): This line performs the K-Means clustering on the scaled_features data and assigns the predicted cluster labels to a new column called 'cluster' in the movie_data.

Hierarchical clustering: This code implements the Hierarchical clustering algorithm to the data, creates 20 clusters using the Ward and average linkage method and Euclidean distance, and assigns the predicted cluster labels to a new column.

1. hierarchical=AgglomerativeClustering(n_clusters=20, metric='euclidean', linkage='ward' or 'Average'): This line creates an AgglomerativeClustering object with the following parameters:

n_clusters=20: Specifies that the hierarchical clustering algorithm should create 20 clusters.

linkage='ward': Specifies the linkage criterion used to merge clusters. 'ward' minimizes the variance within each cluster.

2. movie_data['cluster_hierarchical'] = hierarchical.fit_predict(scaled_features): This line performs the hierarchical clustering and assigns the predicted cluster labels to a new column.

CONCLUSION

In this project, unsupervised learning techniques, specifically K-means and hierarchical clustering, was applied to the MovieLens dataset to group movies based on their attributes. The objective was to uncover hidden patterns and natural groupings without predefined labels.

The process was initiated by applying the K-means clustering technique to form clusters and made subsequent improvements that were done hierarchically with various linkages. The effectiveness of the clustering methods and the selection of the correct number of clusters were further assessed by calculating the silhouette score of all the models.

Based on the output, when using a hierarchical clustering approach with an “average” linkage method ensures that the clusters gotten are very well separated, and the silhouette scores mean something. This implies that hierarchical clustering was efficient in providing the inherent structure of the movie data set.

In sum, the tools used in the project showed effective learning results on the applied unsupervised clustering tasks, and the findings obtained can be applied in practical projects such as the movie recommendation system in the future.